Comp. Part. Mech. (2014) 1:357–372
DOI 10.1007/s40571-014-0020-9

Optimal parameters for a hierarchical grid data structure for contact detection in arbitrarily polydisperse particle systems

Dinant Krijgsman · Vitaliy Ogarko · Stefan Luding

Received: 10 January 2014 / Revised: 10 April 2014 / Accepted: 14 April 2014 / Published online: 22 May 2014

Abstract The objective of this paper is to find the optimum number of hierarchy levels and their cell sizes for contact detection algorithms based on a versatile hierarchical grid data structure, for polydisperse particle systems with arbitrary distribution of particle radii. These algorithms perform as fast as $O(N)$ for N particles, but the prefactor can be as large as N for a given system, depending on the algorithm parameters chosen, making a recipe for choosing these parameters necessary. We estimate theoretically the calculation time of two distinct algorithms for particle systems with various packing fractions, where the sizes of the particles are modelled by an arbitrary probability density function. We suggest several methods for choosing the number of hierarchy levels and the respective cell sizes, based on truncated power-law radii distributions with different exponents and widths. The theoretical estimations are then compared with simulation results for particle systems with up to one million particles. The proposed recipe for selecting the optimal hierarchical grid parameters allows to find contacts in arbitrarily polydisperse particle systems as fast as the commonly-used linked-cell method in purely monodisperse particle systems, i.e., extra work is avoided in presence of polydispersity. Furthermore, the contact detection time per particle even decreases slightly with increasing polydispersity or decreasing particle packing fraction.

Keywords Contact detection · Collision detection · Hierachical grid · Polydisperse · Computational cost · Discrete element

Dinant Krijgsman and Vitaliy Ogarko have contributed equally to this study.

D. Krijgsman (✉)
Univeristy of Twente, PO Box 217, 7500 AE Enschede, The Netherlands
e-mail: d.krijgsman@utwente.nl

V. Ogarko
e-mail: vogarko@gmail.com

S. Luding
e-mail: s.luding@utwente.nl

1 Introduction

Collision detection is a basic computational problem arising in systems that involve spatial interactions among many objects such as particles, granules or atoms in many diverse fields such as robotics, computer graphics, physical simulations, cloth modelling, computational surgery, crowd simulations, etc. All these systems have rather short-ranged interaction in common. Particle based modelling techniques like the discrete element method, (event-driven) molecular dynamics, Monte-Carlo simulations and smoothed particle hydrodynamics, to name a few, play an important role for physically based simulations of powders, granular materials, fluids, colloids, polymers, liquid crystals, proteins and other materials. The performance of the computation relies on several factors, which include the physical model, on the one hand, and the contact detection algorithm used, on the other. The contact detection of pairwise interactions between particles can be one of the most time-consuming tasks in calculations when no suitable contact detection algorithm is used. Because the number of objects treated in simulations is often large, contact detection can become a computational bottleneck. For this reason, the development of efficient contact detection algorithms is crucial to the overall performance of simulations.

With the straightforward "all-to-all" approach each pair of particles is checked for collision. This requires $O(N^2)$ collision checks for N particles, which is computationally prohibitively expensive. More efficient contact detection methods use a two-phase approach to reduce the computational

costs: a broad phase and a narrow phase [16]. The broad phase determines pairs of objects that might possibly collide. It is frequently done by dividing space into regions and testing if objects are close to each other in space. Because objects can only intersect or interact if they occupy the same region of space, the number of pairwise tests can be reduced to $O(N)$. The pairs that "survive" the broad phase test are passed to the narrow phase, which uses specialised techniques to test each candidate pair for a real contact [6,17,34]. The latter is trivial for spherical particles, where one has to compare the distance between particle centres with the sum of particle radii, but can be very costly for particles of arbitrary shape. For example, if there are S surface points per particle, a naive scheme may take order $O(S^2)$ operations. More sophisticated schemes, such as the discrete function representation scheme, require on average $O(S^{1/2})$ operations [34]. Since the broad phase basically acts as a filter for the narrow phase, choices for the two algorithms can usually be made independently.

We distinguish three types of broad phase contact detection methods/data structures: (i) based on coordinate sorting (or spatial sorting), e.g., sweep and prune, (ii) based on Delaunay triangulation, and (iii) based on spatial subdivision, e.g., (hierarchical) grids (or cell-based methods) and trees (e.g., Octrees in $3D$ and Quadtrees in $2D$). Below we briefly describe the above methods and their advantages and weaknesses, while for the detailed analysis see Refs. [4,5,16,18–20,27] and references therein.

Contact detection algorithms based on coordinate sorting imply maintaining particles in a sorted structure along each axis [23,28]. These methods are not sensitive to the particle sizes (i.e., radii for spherical particles) and consume $O(N)$ memory; but they require sorting, which can range in effort from $O(N)$ to $O(N^2)$, depending on the sorting method used, volatility of the sorting lists over time and spatial distribution of objects.

The Delauney triangulation data structure consumes $O(N)$ memory and it is not sensitive to the particle sizes when weighted triangulation is used [25]. However, it has the disadvantage that building (or re-building) the structure has a high computational cost, especially for moving particles. The use of flipping algorithms for maintaining and only incrementally updating the triangulation allows decreasing the overhead of re-building the triangulation [20], but unfortunately in three-dimensional system flipping can get "stuck" [7]. Furthermore, its parallelisation and maintaining of periodic boundary conditions (which are frequently used in particle simulations) is complicated.

The tree data structure for contact detection does not allow to choose cell sizes at every level of hierarchy independently, therefore, leaving no room for optimisation for various distribution of particle sizes [12,31,32]. Moreover, accessing neighbour sub-cubes in the tree is not straightforward since they can be nodes of different tree branches; no more details are given here since this method is not used any further.

The single-level grid-based contact-detection methods, like for example the linked-cell method [1,9,26], are straightforward, widely used and perform well for similarly sized objects. The problem of such methods is their inability to efficiently deal with particles of greatly varying sizes [10]. If the particles within the system are polydisperse, the cell size of a grid would have to conform to the largest particle size. Then many small particles may occupy the same cell, which increases the number of pairwise checks, and therefore affects the computational performance a lot.

This cell size problem can effectively be addressed by the use of multi-level hierarchical grids [3,4,8,10,15,16,22,24]. Particles are positioned at different levels (according to their size) and collision checks are performed in two steps: (i) within the level of insertion (which is usually performed in the same way as in the linked-cell method), and (ii) cross-level checks. The cell size at each hierarchy level can be selected independently, therefore one can adapt grid cells according to a given particle size distribution. Several algorithms based on hierarchical grid data structures were employed, which differ in the way in which the above two steps are implemented.

The hierarchical grid data structure performs $O(N)$ for arbitrary polydisperse systems and uses $O(N)$ memory. This data structure is robust, can be easily parallelised, allows straightforward handling of periodic boundary conditions and can easily deal with unbounded systems. Moreover, it provides $O(1)$ access to the particle data and to all particle nearest neighbours, and, more importantly, allows for $O(1)$ particle insertion and removal from the system, which is often needed in modelling of dynamical systems, like for example hopper or granular flows. Finally, it provides a natural multi-scale framework as particles from different hierarchy levels usually have different physical properties besides their size. For example, small particles are often fast (i.e., have higher velocity/energy) and big ones are slow, so they can have different time scales at different hierarchy levels. These are the reasons why we chose the hierarchical grid as our primary data structure for contact detection and analyse how to optimise it for fastest contact detection in widely polydisperse particle mixtures.

The hierarchical grid data structure has many parameters to configure, i.e., an arbitrary choice of the number of hierarchy levels plus an arbitrary choice of the cell size at every level of hierarchy (the cell sizes are, as convention, increasing with increasing level of hierarchy). The choice of parameters affects the number of contact checks and the overhead of the algorithm, i.e., the number of times the cells are accessed. Due to the many parameters involved, finding the optimal ones, i.e., those which minimize an average number of calculations, T, is a non-trivial problem. This involves multi-

dimensional optimisation where the optimum dimension is unknown. We are not aware of any study where this question was fully addressed, except for the study by Ogarko et al. [22] in which the authors tried to address this problem by providing a hypothesis on the optimal choice of the hierarchical grid parameters, and then comparing their theoretical predictions with the simulation results.

In this study we theoretically analyse the performance complexity of the hierarchical grid data structure for contact detection in polydisperse particulate systems. We provide detailed analysis on the average number of calculations, T, for two distinct algorithms based on the hierarchical grid data structure as applied to polydisperse systems of spherical particles with a power-law distribution of radii, for various power-law exponents, and for various particle volume fractions. We compare several ways (methods) of choosing the hierarchical grid parameters (i.e., the number of levels and the cell sizes at each level) and present the optimal parameter choice. We provide instructions on which hierarchical grid contact detection algorithm should be used and how to choose the optimal parameters for a given arbitrary distribution of particle radii. Finally, we compare our theoretical predictions with simulation results of realistic particle systems.

In the next section we outline the two different algorithms based on the hierarchical grid data structure that are used in this study. We then analyse the performance of the described algorithms and derive general estimates for the number of contact checks per particle in Sect. 3. Section 4 presents the types of particle size distributions considered in this study. In Sect. 5 we introduce several ways of choosing the hierarchical grid parameters and compare their expected performance. For selected parameters these expected performances are also compared with real discrete particle simulations, using the MercuryDPM code (mercurydpm.org) [11,29,30,33]. Finally, the results are summarised and discussed, with some conclusions in Sect. 6.

2 Algorithm

The hierarchical grid (HGrid) algorithm is designed to determine all pairs of particles, in a set of N particles in a d-dimensional Euclidean space, that overlap or interact. The split between local particle geometry and global neighbour searching is achieved through the use of a bounding volume. This way, the contact detection algorithm is able to treat all particle shapes in the same, simplified way. While any bounding volume can be used, the sphere is chosen for this implementation since it is represented simply by a position of its centre $\mathbf{x}_p$ and its radius r_p, and is rotationally invariant. For differently-sized spheres, $r_{\min}$ and $r_{\max}$ denote the minimum and the maximum particle radius, respectively, and $\omega = r_{\max}/r_{\min}$ is the extreme size ratio.

The algorithm consists of two phases. In the first "mapping phase" all the particles are mapped into a hierarchical grid-space (Sect. 2.1). In the second "contact detection phase" (Sect. 2.2) the potential contact partners are determined for every particle in the system. This list of potentially contacting particle pairs is the output of the algorithm. With this list one can perform geometrical intersection tests to check if particles are really in contact, i.e., if they overlap. For spherical particles this can be achieved easily by comparing the distance between two particles with the sum of the radii. For non-spherical particles these tests become more difficult and computational expensive, however, this is beyond the scope of this paper.

Requirements for the algorithm are:

- All pairs of particles that are in contact must be in the list of potential contacts, i.e., the algorithm is not allowed to miss any pair.
- The list of pairs of particles must be unique, i.e., no pair of contacts may appear twice in the list.
- The list of pairs of particles should be as small as possible.
- The computational time of the algorithm should be as small as possible, and for large N, thus must scale linearly with the number of particles, i.e., $O(N)$.
- The memory consumption of the algorithm must be proportional to the number of particles, i.e., $O(N)$.

2.1 Mapping phase

The d-dimensional HGrid is a set of L regular grids with different cell sizes. Every regular grid is associated with a hierarchy level $h \in [1, L]$, where L is the integer number of hierarchy levels. Each level h has a different cell size $s_h \in \mathbb{R}$, where the cells are d-dimensional cubes. Grids are ordered with increasing cell size such that $h = 1$ corresponds to the grid with smallest cell size, i.e., $s_h < s_{h+1}$. For a given number of levels and corresponding cell sizes, the hierarchical grid-cells are defined by the spatial mapping, M, of points $\mathbf{x} \in \mathbb{R}^d$ to a cell at specified level h:

$$M : (\mathbf{x}, h) \mapsto \mathbf{c} = (\lfloor x_1/s_h \rfloor, ..., \lfloor x_d/s_h \rfloor, h), \quad (1)$$

where $\lfloor x \rfloor$ denotes the floor function.[1] The first d components of a $(d+1)$-dimensional vector $\mathbf{c}$ represent cell indices (integers), and the last one is the associated hierarchy level.

It must be noted that the cell size of each level can be set independently, in contrast to contact detection methods which use a tree structure for partitioning the domain [4,31,32], where the cell sizes are taken as double (or triple) the size of the previous lower level of hierarchy, hence

[1] The largest integer not greater than x.

$s_{h+1} = 2s_h$ (or $3s_h$). The flexibility of independent s_h allows one to select the optimal cell sizes, according to the particle size distribution, to improve the performance of the contact detection algorithm.

Using the mapping M, every particle p can be mapped to its cell:

$$\mathbf{c}_p = M(\mathbf{x}_p, h(p)), \tag{2}$$

where $h(p)$ is the *level of insertion* to which particle p is mapped to. The level of insertion $h(p)$ is the lowest level, where the cells are big enough to contain the particle p:

$$h(p) = \left\{ \min_{1 \le h \le L} h : 2r_p \le s_h \right\}. \tag{3}$$

In this way the diameter of particle p is smaller or equal to the cell size at the level of insertion and therefore the classical linked-cell method [1] can be used to detect contacts among particles within the same level of hierarchy.

Figure 1 illustrates a 2-dimensional two-level grid for the special case of a bi-disperse system with $r_{\min} = 3/2$, size ratio $\omega = 8/3$, and cell sizes $s_1 = 3$, and $s_2 = 8$. Since the system contains particles of only two different sizes, two hierarchy levels are sufficient here.

2.2 Contact detection phase

After all particles are mapped to their cells, the contact detection phase is able to calculate all potential contacts. The contact detection is performed by looping over all particles p and searching for possible contacts with particles at the same hierarchy level h and for possible contacts at different hierarchy levels.

Searching for contacts at the same hierarchy level is performed using the classical linked-cell method [1]. The search is done in the cell where p is mapped to, i.e. $\mathbf{c}_p$, and in its neighbouring (surrounding) cells. Only half of the surrounding cells are searched, to avoid testing the same particle pair twice.

Searching for contacts at other hierarchy levels can be performed in two ways. The first one is the Top-Down method, illustrated in Fig. 1(top). In this method one searches for potential contacts only at levels j lower than the level of insertion: $1 \le j < h$. This implies that the particle p will be checked only against smaller particles, thus avoiding double checks for the same pair of particles. The second method, the Bottom-Up method, sketched in Fig. 1(bottom), does exactly the opposite. Here potential contacts are only searched for at hierarchy levels j higher than the level of insertion: $h < j \le L$. This implies that the particle p will be checked only against larger particles, thus avoiding double checks for the same pair of particles.

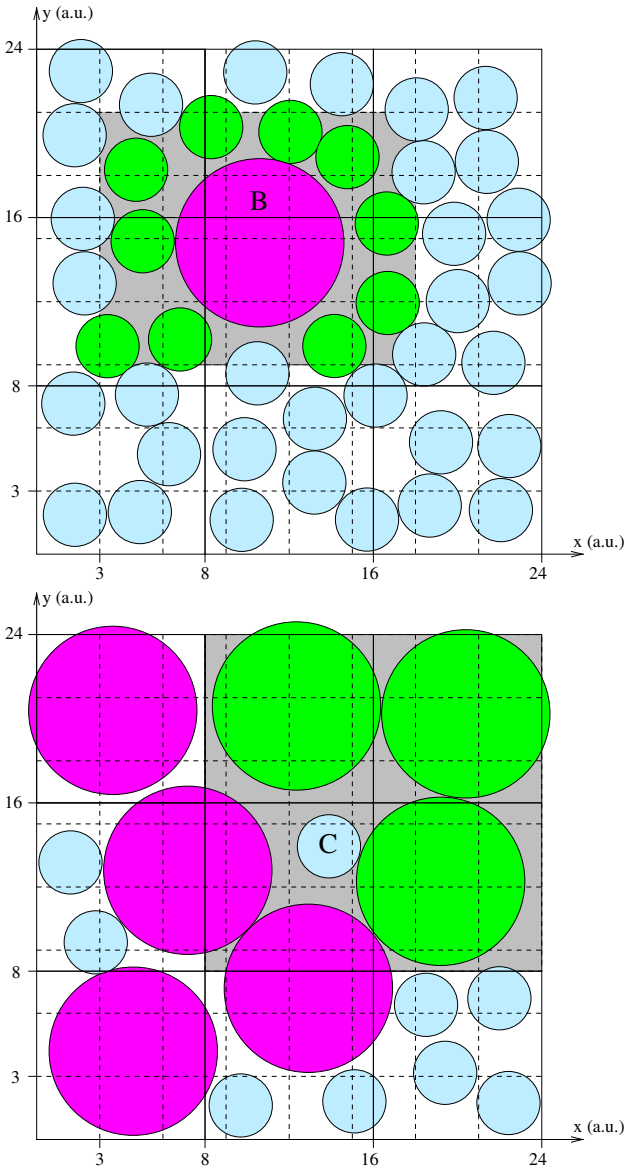


Fig. 1 A 2-dimensional two-level grid for the special case of a bi-disperse system with cell sizes $s_1 = 2r_{\min} = 3$ (a.u.), and $s_2 = 2r_{\max} = 8$ (a.u.). The first level grid is plotted with *dashed lines* while the second level is plotted with *solid lines*. (*Top*) The Top-Down case: The radius of particle B is $r_B = 4$ (a.u.) and its position is $\mathbf{x}_B = (10.3, 14.4)$. Therefore, according to Eqs. (2) and (3), particle B is mapped to the second level to the cell $\mathbf{c}_B = (1, 1, 2)$. The cross-level cells that have to be checked for possible contacts with particle B range from (1,3,1) to (5,6,1), and are marked in *grey*. (*Bottom*) The Bottom-Up case: A particle C is mapped to the cell $\mathbf{c}_C = (4, 4, 1)$. The cross-level cells that have to be checked for possible contacts with particle C range from (1, 1, 2) to (2, 2, 2), and are marked in *grey*. The particles located in the marked (*grey*) cells are coloured dark (*green*). (Color figure online)

The details for both methods are actually quite similar. The algorithm to find potential contacts for particle p at hierarchy level h with other particles at hierarchy level j is as follows:

(1) Define the cells $\mathbf{c}^{\text{start}}$ and $\mathbf{c}^{\text{end}}$ at level j as

$$\mathbf{c}^{\text{start}} := M(\mathbf{x}_c^-, j), \quad \text{and} \quad \mathbf{c}^{\text{end}} := M(\mathbf{x}_c^+, j), \tag{4}$$

where a search box (cube in 3D) is defined by $\mathbf{x}_c^{\pm} = \mathbf{x}_p \pm \beta \sum_{i=1}^{d} \mathbf{e}_i$, with $\beta = r_p + s_j/2$ and $\mathbf{e}_i$ is the standard basis for $\mathbb{R}^d$. Any particle q from level j, with centre $\mathbf{x}_q$ outside this box can not be in contact with particle p, since the diameter of the largest particle at this level can not exceed s_j.

(2) The search for potential contacts is performed in every cell $\mathbf{c} = (c_1, \ldots, c_d, j)$ for which

$$c_i^{\text{start}} \le c_i \le c_i^{\text{end}} \quad \text{for all } i \in [1, d], \tag{5}$$

where c_i denotes the i-th component of vector $\mathbf{c}$. In other words, each particle which was mapped to one of these cells is tested for contact with particle p.

In the Top-Down method, the small cells, defined in Eq. (5), which are almost fully covered by big particles (i.e., no small particles can reside in those cells) can be excluded from the contact search, like for example the cells (3, 4, 1) and (3, 5, 1) in Fig. 1(top). However, we do not know how to identify such cells efficiently, and therefore, have not implemented this optimisation.

3 Performance analysis

The algorithm is applicable to arbitrary systems, however, to estimate the performance of the algorithm, we restrict ourself to systems that are homogeneous in time and in space. In such a system accurate estimates can be obtained and optimal HGrid parameters can be found theoretically.

To analyse the algorithm two time consuming effects are considered:

(1) T^{cd} (collision detection effort) The number of possible contacts that have to be examined more closely. The output of the HGrid-algorithm is a number of possible contacts. Optimum HGrid parameters lead to a low number of possible contacts, because for all these possible contacts a computationally expensive exact geometrical intersection test has to be performed to check if the particles really are in contact.

(2) T^{ca} (cells access effort) The number of times information is retrieved from a cell. While the goal of the HGrid is to obtain a list of all possible contacts, it comes at a (computational) cost. This cost is estimated by the number of times information is obtained from a single cell.

To calculate estimates for T^{cd} and T^{ca}, consider a system of N polydisperse particles with:

- Random positions $\mathbf{x}_p$ within a d-dimensional box at packing fraction ν (without excluded volume effects).
- Random radii between $r_{\min}$ and $r_{\max} = \omega r_{\min}$, according to a normalised probability density function $f(r)$ (for more details see Sect. 4).

With these properties the expected mean volume per particle V_p can be calculated using:

$$V_p = V_d \int_{r_{\min}}^{r_{\max}} r^d f(r)\, dr, \tag{6}$$

where V_d is the volume of a d-sphere of unit radius, i.e., $V_2 = \pi$ and $V_3 = (4/3)\pi$. So the total volume V of all particles becomes:

$$V = N V_p. \tag{7}$$

Given this volume and the packing fraction ν, the size A of a d-dimensional box can be calculated as:

$$A = \left(\frac{V}{\nu}\right)^{\frac{1}{d}}. \tag{8}$$

Now define N_h as the expected number of particles at level h and N_h^c as the number of cells at this level:

$$N_h = N \int_{\frac{1}{2}s_{h-1}}^{\frac{1}{2}s_h} f(r)\, dr, \tag{9}$$

$$N_h^c = \left(\frac{A}{s_h}\right)^d = N \frac{V_p}{\nu} \left(\frac{1}{s_h}\right)^d, \tag{10}$$

where $s_0 = 2r_{\min}$ and $s_L = 2r_{\max}$. So the expected average number of particles per cell at level h, m_h, becomes:

$$m_h = \frac{N_h}{N_h^c} = \frac{\nu s_h^d}{V_p} \int_{\frac{1}{2}s_{h-1}}^{\frac{1}{2}s_h} f(r)\, dr. \tag{11}$$

It must be noted that the number of particles per cell m_h is independent of the total number of particles N.

As described in Sect. 2.2, the algorithm checks for possible contacts at the level of insertion and at the other levels. For both types of contacts estimates of the number of possible contacts and the number of cells that have to be accessed are made in the following two subsections.

3.1 Level-of-insertion search

At the level of insertion h, N_h particles are randomly distributed over N_h^c cells. Therefore, the number of particles in a specific cell at this level, X_h, is binomially distributed with N_h the number of trials and $1/N_h^c$ the probability of success. With this assumption the expected number of potential contacts within a single cell is obtained (see "Estimated number of contacts within a cell" in Appendix). The number of cells that have to be processed is just equal to the number of cells at this level. Therefore, we obtain:

$$T_h^{cd1} = \frac{1}{2} N_h \frac{N_h - 1}{N_h^c} = \frac{1}{2} m_h \left(N_h - 1\right) \approx \frac{1}{2} m_h N_h, \tag{12}$$

$$T_h^{ca1} = N_h, \tag{13}$$

where in the last step of Eq. (12) it is assumed that the number of particles at level h is much greater than unity. For possible contacts between neighbouring cells one just has to square the expected numbers of particles in a cell, m_h, and multiply it by the number of neighbouring cells that have to be checked, n_c, and the total number of cells at the current level, N_h^c:

$$T_h^{cd2} = n_c N_h^c m_h^2 = n_c N_h m_h, \tag{14}$$

$$T_h^{ca2} = n_c N_h^c m_h = n_c N_h, \tag{15}$$

with $n_c = \frac{1}{2}\left(3^d - 1\right)$, i.e., $n_c = 4$ in 2D and $n_c = 13$ in 3D. We obtain that T_h^{cd1}, T_h^{ca1}, T_h^{cd2} and T_h^{ca2} are all linearly dependent on the number of particles N_h at level h.

3.2 Cross-level search

To estimate the number of potential contacts for the cross-level search, first an estimate of the number of cross-cell checks between particles at hierarchy level $j \neq h$ with particles at level h has to be made. In "Number of cells for cross-level search" in Appendix, the number of cells at level j that have to be scanned for potential contacts with particles at level h is found to be:

$$b\left(j,h\right) = \frac{\int_{\frac{1}{2}s_{h-1}}^{\frac{1}{2}s_h} \left(2\frac{r}{s_j} + 2\right)^d f\left(r\right) dr}{\int_{\frac{1}{2}s_{h-1}}^{\frac{1}{2}s_h} f\left(r\right) dr}, \tag{16}$$

with (expected) lower and upper limits:

$$b_{lower}\left(j,h\right) = \left(2 + \frac{s_{h-1}}{s_j}\right)^d, \tag{17}$$

$$b_{upper}\left(j,h\right) = \left(2 + \frac{s_h}{s_j}\right)^d. \tag{18}$$

The expected number of cross-level checks and the number of cells that have to be accessed within a cross-level check can easily be calculated.
For the Top-Down algorithm:

$$T_h^{cd3} = N_h \sum_{j=1}^{h-1} m_j b\left(j,h\right), \tag{19}$$

$$T_h^{ca3} = N_h \sum_{j=1}^{h-1} b\left(j,h\right), \tag{20}$$

and for the Bottom-Up algorithm:

$$T_h^{cd3} = N_h \sum_{j=h+1}^{L} m_j b\left(j,h\right), \tag{21}$$

$$T_h^{ca3} = N_h \sum_{j=h+1}^{L} b\left(j,h\right). \tag{22}$$

Just as for the level-of-insertion search, we obtain that T_h^{cd3} and T_h^{ca3} are linearly dependent on the number of particles N_h at level h, for both algorithms.

3.3 Total computational work

The total computational work per level can now be calculated by just summing of its components.
For the Top-Down algorithm:

$$T_h^{cd} = N_h \left(\left(\frac{1}{2} + n_c\right) m_h + \sum_{j=1}^{h-1} m_j b\left(j,h\right) \right), \tag{23}$$

$$T_h^{ca} = N_h \left(1 + n_c + \sum_{j=1}^{h-1} b\left(j,h\right) \right), \tag{24}$$

and for the Bottom-Up algorithm:

$$T_h^{cd} = N_h \left(\left(\frac{1}{2} + n_c\right) m_h + \sum_{j=h+1}^{L} m_j b\left(j,h\right) \right), \tag{25}$$

$$T_h^{ca} = N_h \left(1 + n_c + \sum_{j=h+1}^{L} b\left(j,h\right) \right). \tag{26}$$

Note that both T_h^{cd} and T_h^{ca} are linear in the expected number of particles at level h, N_h, for both methods, because it was shown in Eq. (11) that m_h is independent of the number of particles. This means that the complexity of the total algorithm is linearly dependent on the total number of particles N, for any number of levels L used. However, depending on

the packing fraction and the particle radii distribution function a huge pre-factor in front of N, even larger than N, can appear when choosing inappropriate HGrid parameters (i.e., cell sizes and number of hierarchy levels).

To find the optimal number of hierarchy levels and their cell sizes, an estimate of the required computational time that is associated with both types of effects, i.e., collision detection work and cell access work, is required. Therefore, the ratio K of time required for a single geometric contact detection over the time required to retrieve information from a cell is introduced. From simulations it is found to be close to $K = 0.2$ for spherical particles and not dependent on the particle volume fraction [21]. Therefore, an estimate of the total time required for a contact detection step is found to be:

$$T = \sum_{h=1}^{L} \left(T_h^{cd} + K T_h^{ca} \right). \tag{27}$$

This result is general in the sense that it describes every possible particle system. However, to get a feel for the optimal HGrid parameters, we limit ourself to a single type of particle size probability distribution function.

4 Particle size probability distribution functions

In order to estimate the performance of the HGrid-algorithm, the distribution of particle radii has to be known. To account for all possible radii distributions the previous section used a normalised probability distribution function $f(r)$. The probability to find a particle with radius between r and $r + dr$ is equal to $f(r)\,dr$. This requires that:

$$\int_0^\infty f(r)\,dr = 1. \tag{28}$$

No general strategy has been found to determine the optimal HGrid-parameters for a general particle radii probability distribution function. Therefore, throughout the remainder of this paper a (truncated) power law size distribution with a constant exponent α is used:

$$f(r) = Cr^\alpha \quad \text{for} \quad r_{\min} \le r \le r_{\max}, \tag{29}$$

with normalisation factor

$$\frac{1}{C} = \int_{r_{\min}}^{r_{\max}} r^\alpha dr. \tag{30}$$

Different values of α have different physical significance, for three-dimensional systems they represent (see also Fig. 2):

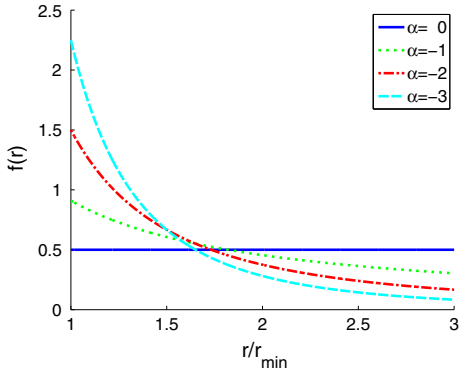


Fig. 2 Probability density function, Eq. (29), for different values of α with $\omega = r_{\max}/r_{\min} = 3$

- $\alpha = 0$: Uniform size (rectangular) distribution, i.e., same number of bigger as smaller particles in intervals dr.
- $\alpha = -2$: Uniform area distribution, i.e., the total surface area of particles with radii between r_1 and $r_1 + dr$ is equal to the total surface area of particles with radii between r_2 and $r_2 + dr$, etc.
- $\alpha = -3$: Uniform volume distribution, i.e., the total volume occupied by particles with radii between r_1 and $r_1 + dr$ is equal to the total volume occupied by particles with radii between r_2 and $r_2 + dr$, etc.

In general, $\alpha > 0$ (not used further) implies that there are more bigger particles, whereas $\alpha < 0$ implies that there are more smaller particles, while $\alpha = 0$ corresponds to a similar number of small and big particles.

5 Cell sizes distribution

Having defined the HGrid algorithm, the last thing to do is to decide about the number of hierarchy levels L and the sizes associated with these levels s_h. In this section four different cell size distributions are introduced, discussed and compared in order to find optimal HGrid parameters. In the end, the predicted performance of the algorithm is compared against real discrete particle method (DPM) simulations, using MercuryDPM (mercurydpm.org) [29,33].

5.1 Single-level grid

As a simple reference, we consider the case of a single hierarchy level ($L = 1$), i.e., the linked-cell method, and compute the total work T as a function of volume fraction ν, size ratio ω, exponent of the size distribution α and dimension d. Due to $L = 1$, we have $N_1 = N$ and $s_1 = s_L = 2r_{\max} = 2\omega r_{\min}$. Using the definition of the number of particles per cell m_h from Eq. (11) and inserting the average particle volume of

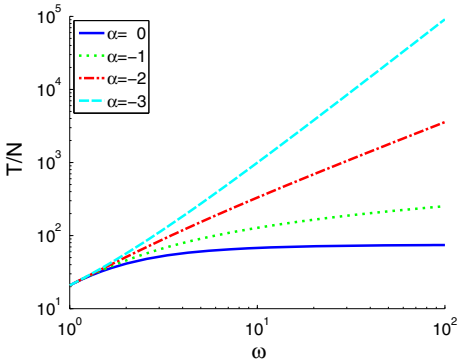


Fig. 3 Computational effort of the HGrid algorithm using a single-level grid (i.e., the linked-cell method) as a function of the width of the particle size distribution, ω, for various exponents α using $d = 3$, $v = 0.7$ and $K = 0.2$

Eq. (6) we obtain:

$$m_1 = (2\omega r_{\min})^d \frac{v}{V_d} \frac{\int_{r_{\min}}^{\omega r_{\min}} f(r)\, dr}{\int_{r_{\min}}^{\omega r_{\min}} r^d f(r)\, dr}. \tag{31}$$

Now substituting the particle radii probability function of Eq. (29) and evaluating the integrals yields (for $\omega \neq 1$, $\alpha \neq -1$ and $\alpha \neq -1-d$):

$$m_1 = (2\omega)^d \frac{v}{V_d} \frac{1+d+\alpha}{1+\alpha} \frac{\omega^{1+\alpha}-1}{\omega^{1+d+\alpha}-1}. \tag{32}$$

We are interested in what happens when the polydispersity ω increases, and thus take the limit of ω going to infinity:

$$\lim_{\omega\to\infty} m_1 = \begin{cases} 2^d \frac{v}{V_d} \frac{1+d+\alpha}{1+\alpha} \omega^d & \alpha < -1-d \\ -2^d \frac{v}{V_d} \frac{1+d+\alpha}{1+\alpha} \omega^{-1-\alpha} & -1-d < \alpha < -1 \\ 2^d \frac{v}{V_d} \frac{1+d+\alpha}{1+\alpha} & \alpha > -1 \end{cases}. \tag{33}$$

Equation (33) shows that for $\alpha < -1$ the number of particles per cell m_1 increases with increasing ω. This means that the efficiency of the linked-cell algorithm is heavily dependent on ω. This result is also shown in Fig. 3, where the required computational effort per particle is plotted as a function of the width of the distribution function ω, for different exponents α. All curves, except the one for $\alpha = 0$ diverge. In general this is true for $\alpha > -1$, meaning that the single-level approach is only appropriate for these values of α. In the following subsections different distributions of the HGrid cell sizes using multiple hierarchy levels are tested to find parameters that lead to minimal computation effort.

5.2 Multi-level cell size distribution

5.2.1 Linear cell size distribution

The easiest method to define the HGrid cell sizes is to use a linear distribution:

$$s_h = 2r_{\min}\left(1 + h\frac{\omega-1}{L}\right). \tag{34}$$

Using this cell size distribution the total work T as a function of the number of HGrid levels L can be calculated. The number of levels where the required computational effort is minimal is chosen as the optimal level and is denoted by L^*. The minimal work T and the optimal number of levels L^* are shown in Fig. 4 for uniform size ($\alpha = 0$) and uniform volume ($\alpha = -3$) particle size distributions. Comparing the work in Fig. 4a with the work for the linked-cell method (Fig. 3), it becomes immediately clear that the HGrid algorithm reduces the work significantly. For $\alpha = 0$ the improvement is less significant, but still the computational effort is reduced by approximately 60 %, while for $\alpha = -3$ a speed-up of several orders of magnitude is achieved. Furthermore, while for $\alpha > 0$ the Bottom-Up algorithm works slightly better, for $\alpha < 0$ the Top-Down algorithm is preferred (data for other values of α not shown). However, in Fig. 4b the disadvantage of using a linear cell size distribution becomes clear. For the $\alpha = -3$ case, the optimal number of levels increases significantly with increasing ω. Therefore, a different cell size distribution might give better results, as we show in the following subsections.

5.2.2 Exponential cell size distribution

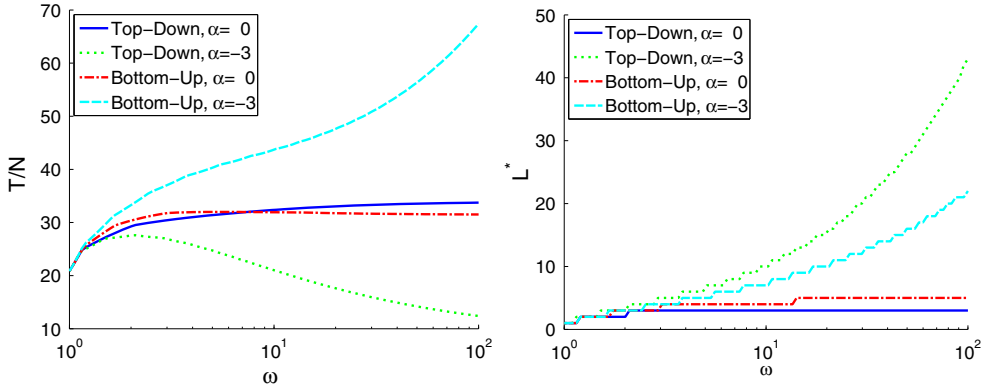
To reduce the optimal number of required levels for the HGrid algorithm an exponential cell size distribution is tested. This distribution stems from the hierarchical tree data structure, where the cell sizes are usually taken as double the size of the previous lower level of hierarchy. This can be generalised by taking cell sizes which are defined as:

$$s_{h+1} = q s_h, \tag{35}$$

with $q > 1$, to make sure that higher level cell sizes are larger. If one substitutes the boundary conditions ($s_0 = r_{\min}$ and $s_L = \omega r_{\min}$), the system of equations can be solved analytically:

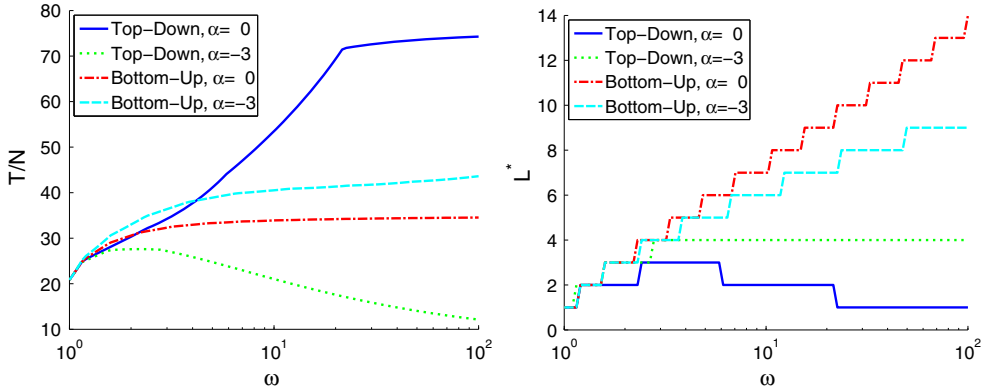
$$s_h = 2r_{\min}\omega^{\frac{h}{L}}. \tag{36}$$

As for the linear cell size distribution, the total work is calculated as a function of the number of HGrid levels and the optimum values are selected and plotted in Fig. 5 for differ-



(a) Computational effort. **(b)** Number of levels.

Fig. 4 Computational effort and optimal number of levels for the HGrid algorithm with a linear cell size distribution as a function of the width of the particle size distribution, ω, for various exponents α, for both the Top-Down and the Bottom-Up algorithms using $d = 3$, $\nu = 0.7$ and $K = 0.2$



(a) Computational effort. **(b)** Number of levels.

Fig. 5 Computational effort and optimal number of levels for the HGrid algorithm with an exponential cell size distribution as a function of the width of the particle size distribution, ω, for various exponents α, for both the Top-Down and the Bottom-Up algorithms using $d = 3$, $\nu = 0.7$ and $K = 0.2$

ent values of α and ω. For $\alpha = 0$ the Bottom-Up algorithm works better, while for $\alpha = -3$ the Top-Down algorithm is preferred. Data for other values of α (not shown here), indicate that in general for $\alpha \leq -2$ the Top-Down algorithm is preferred. The optimal number of levels is not that strongly dependent on ω as the linear cell size distribution, when the Top-Down approach is used. However, there is still a trend: the optimal number of levels increases with increasing ω for the Bottom-Up algorithm.

5.2.3 Constant ratio of the number of particles per cell

Another approach originates from the idea that it may be beneficial to keep the number of particles per cell fixed at every hierarchy level [22]. This simple idea can easily be extended to a rule where the ratio of particles per cell over two adjacent levels is fixed:

$$m_{h+1} = q m_h. \tag{37}$$

This implies that for $q < 1$ the number of particles per cell is decreasing, for $q > 1$ increasing and for $q = 1$ constant, with increasing hierarchy level. Using the definition of m_h from Eq. (11) we can rewrite Eq. (37):

$$s_{h+1}^d F(r) \Big|_{s_h/2}^{s_{h+1}/2} = q s_h^d F(r) \Big|_{s_{h-1}/2}^{s_h/2}, \tag{38}$$

where

$$\frac{dF(r)}{dr} = f(r), \tag{39}$$

$s_0 = r_{\min}$ and $s_L = \omega r_{\min}$. This system of equations can be (at least numerically) solved in terms of s_h, once a size distribution function $f(r)$ is specified.

Using this cell size distribution we reduce the problem of selecting the number of levels and their sizes to just choosing the number of levels L and the ratio of particles per cell for different hierarchy levels q. In Ref. [22] the hypothesis was that it is optimal to keep the number of particles per cell at each level constant, or equivalently using $q = 1$. To check

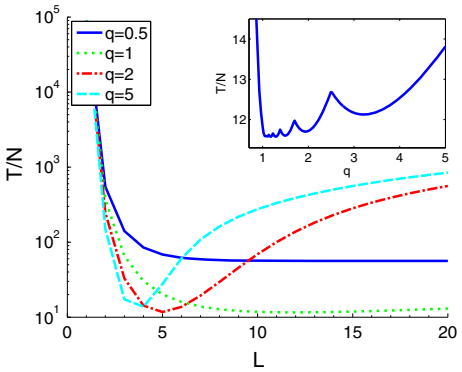


Fig. 6 Computational effort of the HGrid algorithm with a cell size distribution where the ratio of the numbers of particles per cell, q, is constant as a function of the width of the particle size distribution, ω, for different numbers of levels L for the Top-Down method using $\omega = 100, \alpha = -3, d = 3, \nu = 0.7, K = 0.2$. In the *inset* the minimum computational effort is shown for different values of q

this hypothesis the computational effort for different values of q as a function of the number of levels L is shown in Fig. 6 for a system with uniform volume radii distribution ($\alpha = -3$) using $\omega = 100$, $\nu = 0.7$, $d = 3$ and $K = 0.2$. For values of $q = 1$, 2 and 5 the minima in the required computation effort are roughly equal. This is more clearly visible in the inset, where the minimum computational effort is plotted against q. The optimal value is somewhere between $q = 1$ and $q = 2$. The range and number of different levels, for which the computational effort is acceptable, are much bigger for $q = 1$ than for $q = 2$ and thus it is advised to use $q = 1$, for the sake of simplicity. This is also confirmed for different system parameters and the Bottom-Up algorithm (data not shown).

Using $q = 1$, the optimal work and optimal number of levels L^* is shown in Fig. 7. For all values of α the Top-Down algorithm is preferred (data for other values of α not shown).

5.2.4 Optimal cell size distribution

In order to check if the constant number of particles per cell method indeed gives a close-to-optimal result, a numerical optimisation method is used to minimize Eq. (27) in terms of L and s_h under the conditions that $s_{h+1} \geq s_h$, $s_0 = r_{\min}$ and $s_L = \omega r_{\min}$. This is performed using the MATLAB [14] iterative optimisation function "fmincon". In this function, a quadratic programming subproblem is solved at each iteration, where the Hessian of the Langrangian at each iteration is calculated using the BFGS algorithm.

The minimal required computational effort T and the optimal number of levels L^* for this method are shown in Fig. 8. Note that the results are quite comparable to that of the constant number of particles per cell method in Fig. 7.

5.3 Comparison of the cell size distribution functions

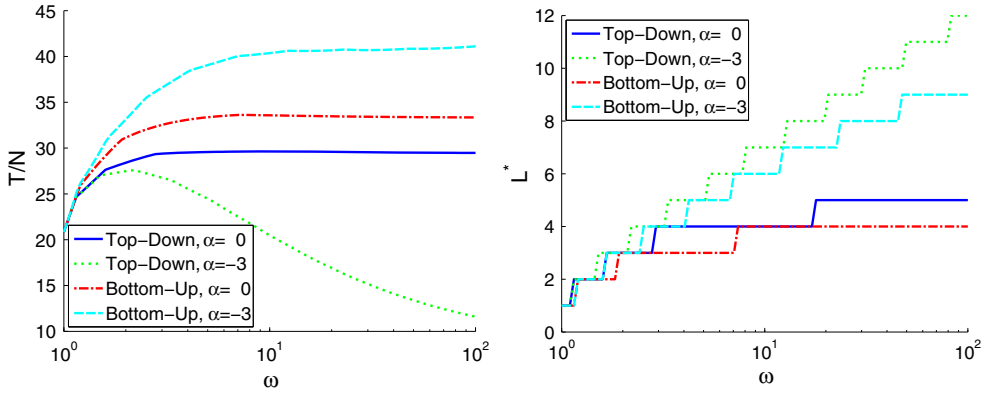
In this subsection the four different cell size distributions are compared and best practices are given. From Fig. 3 and the analysis of the single-level reference case it becomes clear that the HGrid algorithm is essential for $\alpha \leq -1$, however, the required optimal parameters are yet to be determined. The required computational effort for different particle distributions using the four cell size distribution functions is shown in Figs. 4, 5, 6, 7, 8. All of the used algorithms show a significant decrease in computational effort over the single-level reference case for all parameters of the particle size distribution function. Even more important, all but one (the Bottom-Up algorithm using a linear cell size distribution for $\alpha = 0$) of the test cases show that for large polydispersities (i.e., high values of ω) the optimal efficiency of the algorithm is independent on ω. Also the choice of the cell size distribution functions is not too important as long as the other HGrid parameters are chosen optimally. However, in practice it is often difficult or impossible to calculate optimal parameters in advance, due to changing particles, density or geometries. Therefore, the sensitivity of the algorithm to different parameters becomes important.

The required work for all previously discussed cell size distributions is shown in Fig. 9 for the case $\omega = 100$ and $\alpha = -3$ using the Top-Down algorithm. Again we see clearly that the minima of the four curves are roughly equal (12.40, 11.57, 11.60 and 11, 58 respectively), however, the location of the minimum L^* and the sensitivity of the work to using suboptimal parameters differ quite a lot. For the linear cell size distribution the location of the minimum is at $L^* = 43$ (outside the domain of the figure), which is significantly higher than for the other distributions. Such a high number of levels results in additional overhead, especially for particles with complex geometries, therefore it is not advised to use the linear cell size distribution. For the exponential cell size distribution the minimum is located at $L^* = 4$, however choosing $L = 3$ or $L = 6$ already decreases the performance by 43 and 24 % respectively. For the constant number of particles per cell distribution the optimum is located at $L^* = 12$ and choosing $L = 8$ or $L = 19$ reduces the performance just by 10 %. So, it is advised to either use the constant number of particles per cell or the truly optimal cell size distribution, which is even less sensitive to $L \neq L^*$.

The values of the cell sizes s_h and the numbers of particles per cell m_h for the different cell size distributions are shown in Fig. 10. We observe that in most cases $m_h < 1$ is a good choice.

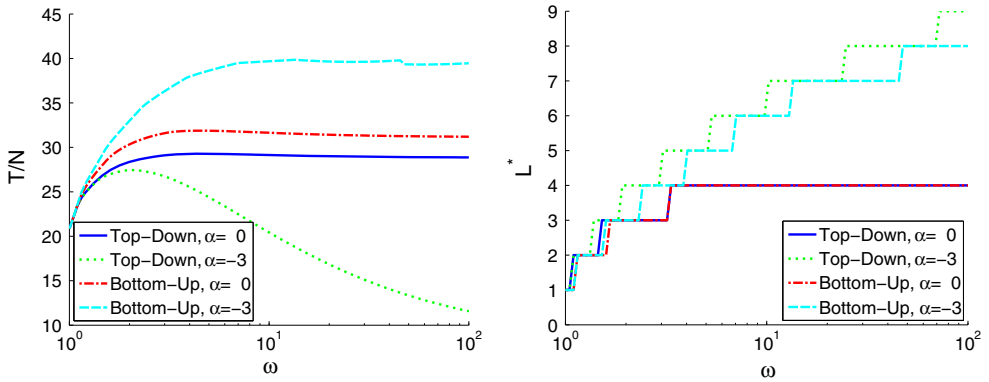
5.4 Comparison with simulations

The estimated computational efficiency of the HGrid algorithm is compared against DPM simulations to check the



(a) Computational effort. **(b)** Number of levels.

Fig. 7 Computational effort and optimal number of levels for the HGrid algorithm with a cell size distribution, where the number of particles per cell is the same at each level as a function of the width of the particle size distribution, ω, for various exponents α, for both the Top-Down and the Bottom-Up algorithms using $d = 3$, $\nu = 0.7$ and $K = 0.2$



(a) Computational effort. **(b)** Number of levels.

Fig. 8 Computational effort and optimal number of levels for the HGrid algorithm with an optimal cell size distribution as a function of the width of the particle size distribution, ω, for various exponents α, for both the Top-Down and the Bottom-Up algorithms, using $d = 3$, $\nu = 0.7$ and $K = 0.2$

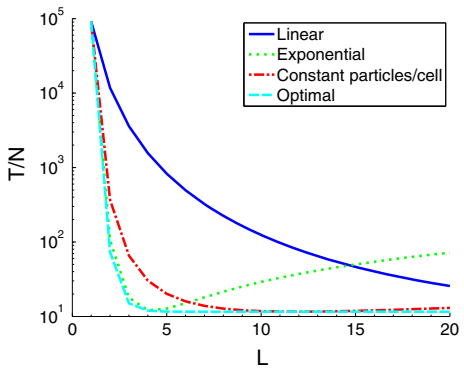
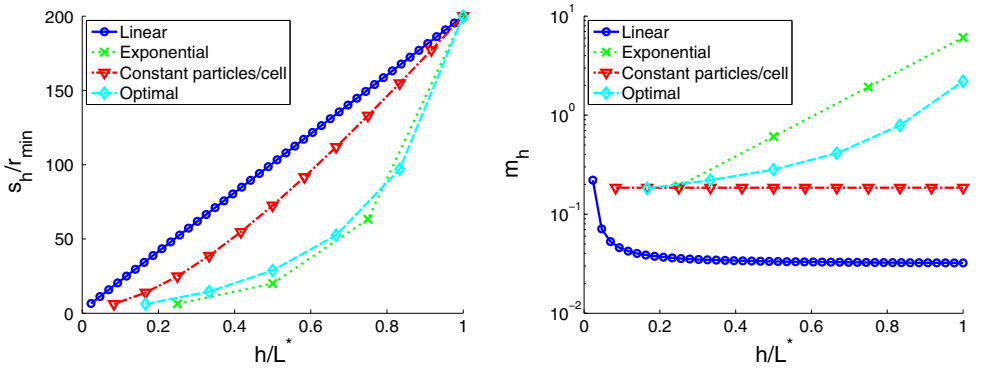


Fig. 9 Computational effort of the HGrid algorithm as a function of the number of levels, L, for different cell size distributions, using $\alpha = -3$, $\omega = 100$, $d = 3$, $\nu = 0.7$ and $K = 0.2$

assumptions used in the derivation. This is done in two steps, first only a single contact detection step has been performed on particle positions obtained from real DPM simulations, while finally a full DPM simulation has been performed with optimal parameters and compared against a simulation using the linked-cell parameters.

5.4.1 Contact detection test

For the single contact detection step, different packings of particles are generated, using a combination of event-driven and soft particle methods, for different packing fractions, particle size distributions and numbers of particles. More specific, we use homogeneous, isotropic, disordered systems of colliding elastic spherical particles in a cubical box with hard walls or periodic boundary conditions (for more details see Ref. [22]). In MercuryDPM (mercurydpm.org) [29,33] a contact detection step has been run using the optimal cell size distribution, where both the number of times a cell is accessed and the number of narrow phase contact detection steps have been counted to compare against the theoretical predictions.



(a) cell sizes. **(b)** Particles per cell.

Fig. 10 Cell sizes and number of particles per cell for the four different cell size distribution methods with optimal HGrid parameters, using $\alpha = -3$, $\omega = 100$, $d = 3$, $\nu = 0.7$ and $K = 0.2$

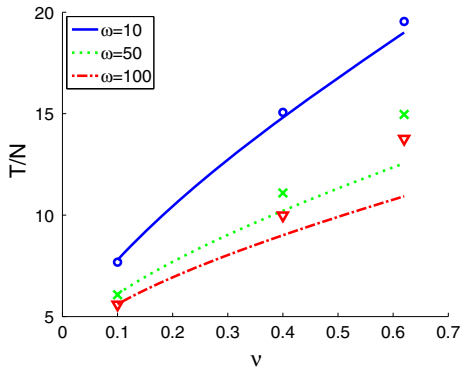


Fig. 11 Comparison of the estimated HGrid computational effort (*lines*) versus that for a real DPM system (*symbols*) for different packing fractions and different polydispersities, using $\alpha = -3$, $N = 1{,}000{,}001$, $d = 3$, $K = 0.2$ and Optimal cell size distribution

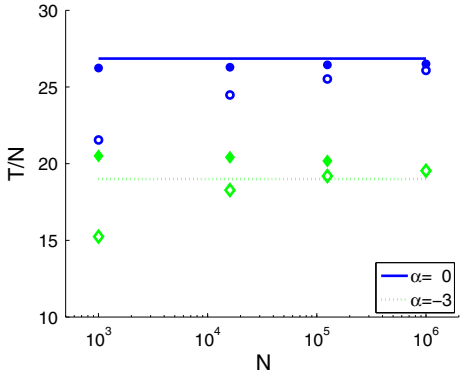


Fig. 12 Comparison of the estimated HGrid computational effort (*lines*) versus that for a DPM system (*symbols*) for different packing fractions and different exponents of the particle size distribution function, using $\omega = 10$, $\nu = 0.62$, $d = 3$, $K = 0.2$ and optimal cell size distribution. *Open symbols* correspond to simulations using solid walls, *filled symbols* represent systems with periodic boundary conditions

In Fig. 11 the results are shown for one million particles using different packing fractions ν and different widths ω of the particle size distribution function. For lower packing fractions the results are extremely accurate, but for higher volume fractions the required work in real simulations becomes slightly higher than expected, however, the overall trend is captured nicely. The main reasons for this deviation we attribute to the excluded volume and finite size effects. In the model derivation the particle centres are assumed to be randomly distributed throughout the domain, whereas in real DPM simulations particles are not allowed to have large overlaps. This is already seen in Fig. 1(top), where large particle B is so big that it completely covers the small grid cells (3, 4, 1) and (3, 5, 1). So the number of cells where the small particles can be distributed is significantly reduced by the presence of large particles. In the test case, for $\alpha = -3$, $\omega = 100$ and $\nu = 0.62$ the percentage of cells on the lowest hierarchy level where two or more particles reside is 2.26 % with excluded volume and 1.26 % for fully random positions. The excluded volume effect leads to a significant increase in the number of cells with two or more particles, and thus also in the number of possible collisions that have to be further examined. This effect increases with increasing volume fraction.

The same conclusion holds for different numbers of particles and different shapes of the particle size distribution function, as shown in Fig. 12. The required computational effort is estimated quite nicely, especially for large numbers of particles. For a small number of particles the computational work is slightly less than expected from the model, because a system of infinite size is assumed. When only a finite number of particles is used there will be particles at the boundary of the domain, which will have less neighbouring particles than particles in the middle of the domain. When using more particles, the ratio of particles at the boundary compared to particles in the central part will become lower and thus increasing the computational effort. This dependence has been tested and confirmed by creating and testing

systems with periodic boundary conditions (solid circles in Fig. 12).

5.4.2 Full DPM test

Throughout this paper the performance of the HGrid algorithm is estimated and measured from the number of times a cell is accessed and the number of narrow phase contact detection steps that are performed. In real DPM simulations, however, additional computational work is required, for example, during the integration routines, for handling periodic boundaries or walls and for writing data. To show the real improvement of using optimal HGrid parameters, a simple free cooling simulation with moderate polydispersity has been run using different HGrid parameters [13]. More specifically, we performed 2D simulations using solid walls with 10^4 particles over 2.5×10^5 time steps. The particle sizes are distributed according to the (truncated) power law size distribution with parameters $\omega = 20$ and $\alpha = -3$, with a packing fraction of 0.4. According to our analysis the optimal parameters for this system are: the Top-Down algorithm with 5 levels with sizes 4.0, 7.9, 15.1, 27.2 and 40 times r_{min}, respectively. These settings should give a speed-up of the contact detection part of the simulation by a factor of 35 when comparing against a linked-cell reference case (i.e., one level with size 40). The full DPM simulation with optimal parameters took about 27 min, whereas the reference case required 266 minutes. This speed-up by a factor of 9.9 is naturally lower than the predicted speed-up of the contact detection part due to force calculations, integration routines and wall interactions, but is still quite significant.

6 Summary and conclusion

Contact detection is a fundamental problem that occurs in many different kinds of simulation methods. This process is often computationally expensive, usually taking up a considerable proportion of CPU time, especially for systems with non-uniform density or polydisperse particle sizes.

In this paper, we studied analytically the computational effort of two algorithms for contact detection (i.e., Bottom-Up and Top-Down), based on the multi-level hierarchical grid data structure. The basic idea of these algorithms is the fact, that usually there are lots of particles in the system, which cannot be in contact, as they are too distant. The presented methods save a lot of time by excluding such particles from a detailed and time consuming contact examination and evaluation. The performance of the neighbour searching algorithm based on both the number of particles and the width of the particle size distribution, is of great importance.

As an input for the algorithm, the number of hierarchy levels and their cell sizes are required. Therefore, we tested four methods for choosing the hierarchical cell size distribution (i.e., linear, exponential, constant number of particles per cell and optimal) and compare their theoretical performance for a power law particle size distribution function with exponent α. For almost all methods the performance of the algorithm becomes independent of the width of the particle size distribution ω, in contrast to the linked-cell method. Even better, the computational effort per particle, using the algorithm decreases with increasing ω, or with decreasing α, at constant system packing fraction. In general, with optimal parameters, the algorithm is able to find contacts in arbitrarily polydisperse particle systems as fast as the linked-cell method finds contacts in purely monodisperse particle systems, i.e., no extra work is required due to polydispersity.

For the linear cell size distribution the optimal number of hierarchy levels is huge for systems with large polydispersity and $\alpha < 0$ (i.e., the systems dominated by small particles). Therefore, for these kinds of systems, the linear cell size distribution has high computational overhead and in general does not perform well, especially for particles with complex geometries. The exponential cell size distribution performs better, however, it is very sensitive to the number of hierarchy levels used. So it is not appropriate to use this method in dynamical systems, where the particle size distribution, density or system geometry is changing over time. Both the constant number of particles per cell and the optimal cell size distribution methods perform well, are not too sensitive to the number of levels, and have low overhead.

For $\alpha \leq -1$ the use of a multilevel grid becomes extremely efficient (i.e., orders of magnitude faster) as compared to the single level linked-cell method, if optimal parameters are used. On the other side, for $\alpha > -1$, the use of a multilevel grid does not present a major advantage (but can improve performance slightly). In all our test cases (with optimum HGrid parameters), the contact detection time is estimated to be $T \leq 30N$ for three-dimensional systems with spherical particles (where a unit of time is defined as the time required for a two-sphere overlap test). For future research, our analysis technique allows to investigate how the algorithm performs for other more realistic size distributions, e.g., log-normal.

7 Recommendations

In this section recommendations are given for setting the HGrid parameters.

- Use the Top-Down algorithm.
- If possible, perform your own minimisation using your exact system and the overhead factor K applicable to

your solver, to obtain the optimal number of levels and their cell size distribution.
- Otherwise, use a cell size distribution where the number of particles per cell is approximately the same at each level.
- Since the optimum number of levels L depends on the particle size distribution function and the packing fraction no general recommendations can be given. For distributions similar to the (truncated) power law size distribution used throughout this paper we refer to Fig. 7.

Acknowledgments We would like to thank A. R. Thornton and S. González for helpful discussions. This research is supported by the Dutch Technology Foundation STW, which is the applied science division of NWO, and the Technology Programme of the Ministry of Economic Affairs, project number STW-MUST 10120, STW-HYDRO 12272, and STW-VICI grant 10828.

Appendix

Estimated number of contacts within a cell

To calculate the estimated number of potential contacts within a cell, we assume that at level h there are N_h particles randomly divided over N_h^c cells, such that the probability that a certain particle goes to a certain cell is $1/N_h^c$. The number of particles in cell i, X_i, follows the binomial distribution with parameters N_h and $1/N_h^c$ such that the probability of finding n particles in cell i is equal to:

$$P\left(X_i = n\right) = \left(N_h n\right)\left(\frac{1}{N_h^c}\right)^n\left(1-\frac{1}{N_h^c}\right)^{N_h-n}, \tag{40}$$

where

$$\left(N_h n\right) = \frac{N_h!}{n!\left(N_h-n\right)!} \tag{41}$$

is the binomial coefficient. If n particles reside in a certain cell, we have to check for $n\left(n-1\right)/2$ potential contacts in that cell and thus we can estimate the number of potential contacts within a single cell by calculating its weighted average. However, we are interested in the potential contacts within cells for the whole hierarchy level h and thus have to multiply this by the number of cells at this level N_h^c to obtain T^{cd1}.

$$T^{cd1} = \frac{N_h^c}{2}\sum_{n=0}^{N_h} n\left(n-1\right)P\left(X_i=n\right). \tag{42}$$

First, note that for $n=0$ and $n=1$ the right-hand side equals zero, and thus we can change the summation domain:

$$T^{cd1} = \frac{N_h^c}{2}\times \sum_{n=2}^{N_h} n\left(n-1\right)\left(N_h n\right)\left(\frac{1}{N_h^c}\right)^n\left(1-\frac{1}{N_h^c}\right)^{N_h-n}. \tag{43}$$

Furthermore, by using the definition of the binomial coefficient we obtain:

$$\left(\frac{N_h}{n}\right) = \left(\frac{N_h-1}{n-1}\right)\frac{N_h}{n}. \tag{44}$$

And we can rewrite Eq. (42) as:

$$T^{cd1} = \frac{N_h^c}{2}\frac{N_h}{N_h^c}\frac{N_h-1}{N_h^c}\times \sum_{n=2}^{N_h}\left(\frac{N_h-2}{n-2}\right)\left(\frac{1}{N_h^c}\right)^{n-2}\left(1-\frac{1}{N_h^c}\right)^{N_h-n}. \tag{45}$$

Substituting $n=a+2$ and $N_h=b+2$ gives

$$\begin{aligned} T^{cd1} &= \frac{N_h^c}{2}\frac{N_h}{N_h^c}\frac{N_h-1}{N_h^c}\sum_{a=0}^{b}\binom{b}{a}\left(\frac{1}{N_h^c}\right)^a\left(1-\frac{1}{N_h^c}\right)^{b-a} \\ &= \frac{N_h^c}{2}\frac{N_h}{N_h^c}\frac{N_h-1}{N_h^c}, \end{aligned} \tag{46}$$

where in the second step the definition of a probability density function is used.

Number of cells for cross-level search

To estimate the number of potential contacts for the cross-level search, first an estimate of the number of cells that have to be checked for possible cross-level contacts has to be made. Therefore, consider a particle p at position $\mathbf{x}_p$ with radius r_p such that it resides at hierarchy level h and we want to calculate the number of cells at hierarchy level j that have to be checked for possible contacts. This can be calculated from Eq. (4), which reads for the x-direction (note that the directions are statistically independent):

$$c_1^{\text{start}} = \left\lfloor\frac{x_p-r_p-s_j/2}{s_j}\right\rfloor = \left\lfloor\frac{x_p-r_p}{s_j}-\frac{1}{2}\right\rfloor, \tag{47}$$

$$c_1^{\text{end}} = \left\lfloor\frac{x_p+r_p+s_j/2}{s_j}\right\rfloor = \left\lfloor\frac{x_p+r_p}{s_j}+\frac{1}{2}\right\rfloor. \tag{48}$$

So the number of cells that have to be checked in the x-direction is, (see also Eq. (5)),

$$a\left(j, x_p, r_p\right) = \left\lfloor \frac{x_p + r_p}{s_j} + \frac{1}{2} \right\rfloor - \left\lfloor \frac{x_p - r_p}{s_j} - \frac{1}{2} \right\rfloor + 1. \tag{49}$$

This result holds for a special particle p at position $\mathbf{x}_p$ with radius r_p at level h. However, we want to calculate the average number of checks. Therefore, we have to integrate $a\left(j, x, r\right)$ over all possible positions x and all possible radii r:

$$b\left(j, h\right) = \frac{\int_{\frac{1}{2}s_{h-1}}^{\frac{1}{2}s_h} \left(\int_{-\infty}^{\infty} a\left(j, x, r\right) g\left(x\right) dx\right)^d f\left(r\right) dr}{\int_{\frac{1}{2}s_{h-1}}^{\frac{1}{2}s_h} \left(\int_{-\infty}^{\infty} g\left(x\right) dx\right)^d f\left(r\right) dr}. \tag{50}$$

where $g\left(x\right)$ and $f\left(r\right)$ are the probability density distribution functions for position and radius, respectively. First, note that Eq. (50) is periodic with respect to the position of the particle (x) with a period s_j. So, without loss of generality, we can assume that x is uniformly randomly distributed between 0 and s_j:

$$b\left(j, h\right) = \frac{\int_{\frac{1}{2}s_{h-1}}^{\frac{1}{2}s_h} \left(\int_0^{s_j} a\left(j, x, r\right) \frac{1}{s_j} dx\right)^d f\left(r\right) dr}{\int_{\frac{1}{2}s_{h-1}}^{\frac{1}{2}s_h} f\left(r\right) dr}. \tag{51}$$

To evaluate this integral we rewrite Eq. (49) by splitting $\frac{r}{s_j} + \frac{1}{2}$ into $a + n$, with $n = \left\lfloor \frac{r}{s_j} + \frac{1}{2} \right\rfloor$ and $0 \leq a < 1$:

$$a\left(j, x, r\right) = \left(\left\lfloor \frac{x}{s_j} + a \right\rfloor - \left\lfloor \frac{x}{s_j} - a \right\rfloor + 2n + 1 \right). \tag{52}$$

Note that the following holds

$$\left\lfloor \frac{x}{s_j} + a \right\rfloor = \begin{cases} 0 & \frac{x}{s_j} < 1 - a \\ 1 & x_p \geq 1 - a \end{cases}, \tag{53}$$

$$\left\lfloor \frac{x}{s_j} - a \right\rfloor = \begin{cases} 0 & \frac{x}{s_j} \geq a \\ -1 & x_p < a \end{cases}, \tag{54}$$

such that we can integrate Eq. (51)

$$b\left(j, h\right) = \frac{\int_{\frac{1}{2}s_{h-1}}^{\frac{1}{2}s_h} \left(2a + 2n + 1\right)^d f\left(r\right) dr}{\int_{\frac{1}{2}s_{h-1}}^{\frac{1}{2}s_h} f\left(r\right) dr}, \tag{55}$$

and substitute $a + n = \frac{r}{s_j} + \frac{1}{2}$ back:

$$b\left(j, h\right) = \frac{\int_{\frac{1}{2}s_{h-1}}^{\frac{1}{2}s_h} \left(2\frac{r}{s_j} + 2\right)^d f\left(r\right) dr}{\int_{\frac{1}{2}s_{h-1}}^{\frac{1}{2}s_h} f\left(r\right) dr}. \tag{56}$$

Without knowing the exact particle size distribution function this integral can not be calculated. However, its upper and lower bounds are:

$$b_{lower}\left(j, h\right) = \left(2 + \frac{s_{h-1}}{s_j}\right)^d, \tag{57}$$

$$b_{upper}\left(j, h\right) = \left(2 + \frac{s_h}{s_j}\right)^d. \tag{58}$$

References

1. Allen MP, Tildesley DJ (1989) Computer simulations of liquids. Clarendon Press, Oxford
2. Donev A, Stillinger FH, Torquato S (2005) Neighbor list collision-driven molecular dynamics simulation for nonspherical particles. I. Algorithmic details. J Comput Phys 202(2):737–764
3. Eitz M, Lixu G (2007) Hierarchical spatial hashing for real-time collision detection. In: Proceedings of the international conference on shape modeling and applications, SMI 2007,Lyon, pp 61–70
4. Ericson C (2004) Real-time collision detection., The Morgan Kaufmann Series in Interactive 3-D TechnologyMorgan Kaufmann Publishers Inc., San Francisco, CA
5. Gavrilova ML, Rokne JG (2002) Collision detection optimization in a multi-particle system. In: Proceedings of the international conference on computational science—part III, ICCS 2002, London, pp 105–114
6. Gilbert E, Johnson D, Keerthi S (1988) A fast procedure for computing the distance between complex objects in three-dimensional space. IEEE J Robot Autom 4(2):193–203
7. Guibas L, Russel D (2004) An empirical comparison of techniques for updating delaunay triangulations. In: Proceedings of the 20th annual symposium on computational geometry, SCG 2004, pp 170–179
8. He K, Dong S, Zhou Z (2007) Multigrid contact detection method. Phys Rev E 75:036710
9. Hockney RW, Eastwood JW (1981) Computer simulation using particles. McGraw-Hill, New York
10. Iwai T, Hong CW, Greil P (1999) Fast particle pair detection algorithms for particle simulations. Int J Mod Phys C 10(5):823–837
11. Krijgsman D, Luding S (2013) 2D cyclic pure shear of granular materials, simulations and model. Proc AIP Conf 1524(1):1226–1229
12. Lu L, Gu Z, Lei K, Wang S, Kase K (2010) An efficient algorithm for detecting particle contact in non-uniform size particulate system. Particuology 8(2):127–132
13. Luding S (2008) Introduction to discrete element methods : basic of contact force models and how to perform the micro-macro transition to continuum theory. Eur J Environ Civil Eng 12(7–8):785–826
14. MATLAB (2009) MATLAB: version 7.9.0. The MathWorks Inc., Natick

15. Mio H, Shimosaka A, Shirakawa Y, Hidaka J (2006) Optimum cell condition for contact detection having a large particle size ratio in the discrete element method. J Chem Eng Jpn 39(4):409–416
16. Mirtich B (1997) Efficient algorithms for two-phase collision detection. Technical Report TR-97-23. Mitsubishi Electric Research Laboratory, Cambridge
17. Mirtich B (1998) V-clip: fast and robust polyhedral collision detection. ACM Trans Graph 17:177–208
18. Munjiza A (2004) The combined finite-discrete element method. Wiley, Chichester
19. Muth B, Müller MK, Eberhard P, Luding S (2007) Collision detection and administration methods for many particles with different sizes. In: Proceedings of the 4th international conference on discrete element methods, DEM 2007, Brisbane
20. Ogarko V, Luding S (2010) Data structures and algorithms for contact detection in numerical simulation of discrete particle systems. In: Proceedings of the 6th world congress on particle technology, WCPT6, Nuremberg
21. Ogarko V, Luding S (2011) A study on the influence of the particle packing fraction on the performance of a multilevel contact detection algorithm. In: Proceedings of the 2nd international conference on particle-based methods—fundamentals and applications, Barcelona, pp 1–7
22. Ogarko V, Luding S (2012) A fast multilevel algorithm for contact detection of arbitrarily polydisperse objects. Comput Phys Commun 183(4):931–936
23. Perkins E, Williams J (2001) A fast contact detection algorithm insensitive to object sizes. Eng Comput 18(1–2):48–61
24. Peters JF, Kala R, Maier RS (2009) A hierarchical search algorithm for discrete element method of greatly differing particle sizes. Eng Comput 26(6):621–634
25. Pournin L (2005) On the behavior of spherical and non-spherical grain assemblies, its modeling and numerical simulation. Ph.D. thesis, EPFL, Lausanne
26. Quentrec D, Brot C (1973) New method for searching for neighbors in molecular dynamics computations. J Comput Phys 13(3):430–432
27. Raschdorf S, Kolonko M (2011) A comparison of data structures for the simulation of polydisperse particle packings. Int J Numer Methods Eng 85:625–639
28. Schinner A (1999) Fast algorithms for the simulation of polygonal particles. Granul Matter 2(1):35–43
29. Thornton A, Weinhart T, Luding S, Bokhove O (2012) Modeling of particle size segregation: calibration using the discrete particle method. Int J Modern Phys C 23(8):1240014
30. Thornton AR, Krijgsman D, te Voortwis A, Ogarko V, Luding S, Fransen R, González S, Bokhove O, Imole O, Weinhart T (2013) A review of recent work on the discrete particle method at the university of twente: an introduction to the open-source package mercurydpm. In: Proceedings of the 6th international conference on discrete element methods, DEM-6 Golden, Denver, CO, pp 50–56
31. Ulrich T (2000) Loose octrees. In: Ulrich M (ed) Game programming gems. Charles River Media, Stamford, pp 444–453
32. Vemuri BC, Cao Y, Chen L (1998) Fast collision detection algorithms with applications to particle flow. Comput Graph Forum 17(2):121–134
33. Weinhart T, Thornton A, Luding S, Bokhove O (2012) From discrete particles to continuum fields near a boundary. Granul Matter 14(2):289–294
34. Williams J, O'Connor R (1995) Linear complexity intersection algorithm for discrete element simulation of arbitrary geometries. Eng Comput 12(2):185–201